Journal of Intelligent Information Systems, 4, 89–108 (1994)

Discovering Dynamics: From Inductive Logic Programming to Machine Discovery

SAŠO DŽEROSKI saso.dzeroski@ijs.si
LJUPČO TODOROVSKI ljupco.todorovski@ijs.si
Institut Jožef Stefan, Jamova 39, 61111 Ljubljana, Slovenia

Editor: Gregory Piatetsky-Shapiro

Abstract. Machine discovery systems help humans to find natural laws from collections of experimentally collected data. Most of the laws found by existing machine discovery systems describe static situations, where a physical system has reached equilibrium. In this paper, we consider the problem of discovering laws that govern the behavior of dynamical systems, i.e., systems that change their state over time. Based on ideas from inductive logic programming and machine discovery, we present two systems, QMN and LAGRANGE, for discovery of qualitative and quantitative laws from quantitative (numerical) descriptions of dynamical system behavior. We illustrate their use by generating a variety of dynamical system models from example behaviors.

Keywords: machine discovery, machine learning, dynamical system identification

1. Introduction

The task of modeling natural and artificial dynamical systems, i.e., systems that change their state over time, is omnipresent. Usually, dynamical systems are described by a set of differential equations which completely specify the rate of change of each of the system variables. To make the modeling task easier, qualitative formalisms, such as QDE (Qualitative Differential Equations) [Kuipers, 1986], use qualitative relationships to describe dependencies among the system variables, which do not necessarily specify the rates of change of the system variables in a complete and precise manner. In QDE, the states of the system variables are described as pairs of qualitative values and directions of change and qualitative relationships are defined over sequences of such state descriptions, i.e., qualitative behaviors.

Considerable effort has been devoted to the problem of automating the task of building qualitative models from example behaviors [Bratko, et al. 1992; Coiera, 1989; Džeroski & Bratko, 1992; Kraan, et al. 1991]. Viewing qualitative models as logic programs and formulating the QDE constraints in logic, Bratko et al. [1992] and Džeroski and Bratko [1992] used systems for inductive synthesis of logic programs (inductive logic programming [Muggleton, 1992; Lavrač & Džeroski, 1994]) to automatically synthesize qualitative models from example behaviors. MISQ [Kraan, et al. 1991], a re-incarnation of GENMODEL [Coiera, 1989], can generate a qualitative model from a numerical trace by translating the numerical trace into a qualitative behavior, from which a qualitative model is then generated.

The task of identification of dynamical systems, as addressed in this paper, is to find a set of laws that describe the dynamics of the system from a given example behavior. More precisely, a set of real-valued system variables is measured at regular intervals over a period of time, as illustrated in Table 1. The laws to be discovered (also called a model of the dynamical system) typically take the form of a set of qualitative or ordinary differential equations. We also refer to this task as the task of discovering dynamics.

Table 1. A behavior trace of a dynamical system.

Time	System variables			
	X_1	X_2	$\dots$	X_n
t_0	x_{10}	x_{20}	$\dots$	x_{n0}
t_0+h	x_{11}	x_{21}	$\dots$	x_{n1}
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
t_0+Nh	x_{1N}	x_{2N}	$\dots$	x_{nN}

The first part of the paper describes QMN (Qualitative Models from Numerical traces), a system that generates qualitative models from numerically described behaviors directly, without translating them to qualitative behaviors. To this end, QMN assumes the original numerical interpretation of the qualitative constraints used in QDE [Kuipers, 1986], which include addition, multiplication and time derivation. A generate and test methodology is used in QMN, similar to the ones used in GENMODEL [Coiera, 1989] and MISQ [Kraan, et al. 1991].

It turns out that there is only a short step from the automatic generation of qualitative differential equations from numerical traces to the generation of ordinary differential equations from numerical traces. The latter can be viewed as the problem of discovering quantitative empirical laws about dynamical systems. Although there exists a variety of systems for discovery of quantitative empirical laws, they have not addressed the problem of discovering dynamics. This problem is addressed by the discovery system LAGRANGE, described in the second part of the paper.

Initially, LAGRANGE was to use a very simple process: give the derivatives of the observed system variables and the system variables themselves, observed over a period of time, to an existing discovery system. The latter would then produce a set of laws (differential and algebraic equations) describing the observed behavior. However, existing discovery systems, such as BACON [Langley, et al. 1987] and FAHRENHEIT [Zytkow & Zhu, 1991], were not suitable for use in LAGRANGE as they ask for additional data or consider dependencies between two variables only.

We thus had to develop a discovery mechanism to be used within LAGRANGE, which does not ask for additional data and considers dependencies among several variables at the same time. It is based on multidimensional linear regression, and introduces new terms by multiplication, following ideas from inductive logic programming and from machine discovery systems.

Section 2 describes QMN, a system that generates qualitative models from numerically described behaviors. The use of QMN is illustrated in Section 3, where its application to the behaviors of two dynamical systems is described. LAGRANGE, a machine discovery system that constructs quantitative models, i.e., sets of algebraic and differential equations, is described in Section 4. The experiments with LAGRANGE, which involve the construction of models for several dynamical systems, are described in Section 5. Related work is discussed in Section 6. Finally, Section 7 concludes and elaborates on several directions for further work.

2. The QMN algorithm

QMN (Qualitative Models from Numerical traces) generates a qualitative model from a numerically described dynamical system behavior. QDE [Kuipers, 1986] constraints on the system variables are used in the generated qualitative model. The constraints are taken from the repertoire given in Table 2.

Table 2. QDE constraints tested in QMN.

QDE Constraint	Meaning
$const(F)$	F is constant over time
$deriv(F_1, F_2)$	F_2 is the time derivative of F_1, i.e., $F_2(t) = \frac{d}{dt}F_1(t)$
$minus(F_1, F_2)$	$F_1 = -F_2$
$M^+(F_1, F_2)$	F_1 monotonically increases with F_2
$M^-(F_1, F_2)$	F_1 monotonically decreases with F_2
$add(F_1, F_2, F_3)$	$F_1 + F_2 = F_3$
$mult(F_1, F_2, F_3)$	$F_1 * F_2 = F_3$

The input to QMN is a behavior trace of a dynamical system, such as the one given in Table 1. In addition, the values of four parameters have to be specified. These are: the order o of the dynamical system (the order of the highest derivative in the dynamics equations), the maximum depth d of new variables introduced by combining old variables, and two tolerances, ϵ and δ, used when testing qualitative constraints. Optionally, the dimensions of the system variables can be specified.

Taking the set of system variables $S = \{X_1, \ldots, X_n\}$, QMN first introduces their time derivatives (up to order o). It then introduces new variables by repeatedly applying the basic arithmetic operations to the variables from S and their time derivatives. Finally, given the set of all (old and new) variables, it generates and tests all possible qualitative constraints over these variables.

The time derivatives of the system variables are introduced by numerical derivation (lines 3-6 of the QMN algorithm), using the formula [Bohte, 1991] (p. 73)

$$\dot{x_i} = \frac{1}{12 \cdot h}(x_{i-2} - 8 \cdot x_{i-1} + 8 \cdot x_{i+1} - x_{i+2})$$

where x_i and $\dot{x_i}$ are the values of variables X and its time derivative $\dot{X} = dX/dt$ at time point $t_0 + ih$. As numerical derivation can lead to large errors, due care should be exercised and derivatives should be measured whenever the measurement error is lower than the corresponding numerical derivation error.

1. **QMN**(S, o, d, ϵ,δ):
2.   **Introduce-time-derivatives**
3. $V := S$
4.   **forall** $v \in S$ **do**
5. $v_0 := v$
6.   **for** $i := 1$ **to** o **do** $V := V \cup \{\frac{d}{dt} v_{i-1}\}$
7.   **Introduce-new-variables**
8. $V_1 := V$
9.   **for** $k := 2$ **to** d **do**
10. $V_k := \emptyset$
11.   **forall** $(v, u) \in V_1 \times V_{k-1}$ **do**
12. $a_{v,u} := v + u$
13. $s_{v,u} := v - u$
14. $m_{v,u} := v * u$
15. $d_{v,u} := v/u$
16. $V_k := V_k \cup \{a_{v,u}, s_{v,u}, m_{v,u}, d_{v,u}\}$
17. $V := V \cup V_k$
18.   **Generate-and-test-qualitative-constraints**
19. $M := \emptyset$
20.   **forall** $v \in V$ **do**
21.   **if Satisfied**($const(v)$, ϵ, δ) **then** $M := M \cup \{const(v)\}$
22.   **forall** $(v, u) \in V \times V$ **do**
23.   **if Satisfied**($deriv/minus/M^+/M^-(v, u)$, ϵ, δ)
24.   **then** $M := M \cup \{deriv/minus/M^+/M^-(v, u)\}$
25.   **forall** $(v, u, w) \in V \times V \times V$ **do**
26.   **if Satisfied**($add/mult(v, u, w)$, ϵ, δ)
27.   **then** $M := M \cup \{add/mult(v, u, w)\}$
28.   **return**(M)

The system variables and their time derivatives are then combined pairwise in all possible ways, using the four basic arithmetic operations (lines 8-17). Only variables of the same dimension may be added and subtracted. The values of the new variables at all time points are calculated as the new variables are introduced. Finally, qualitative constraints are generated and tested (lines 19-27).

The testing of all constraints is based on testing the constraint $zero(T)$, which holds when the variable T represents the constant zero within the tolerances ϵ and δ. More precisely, a variable T is considered to represent the constant zero if $P(|T| > \delta) < \epsilon$. This probability is approximated with the proportion of measurements for which $|T| > \delta$ holds. The parameter δ is, in fact, a numerical precision tolerance, and the parameter ϵ allows the constraint to be considered satisfied even if a small proportion (ϵ) of measurements are not consistent with it.

Testing of the QDE constraints, implemented in the procedure **Satisfied**, is done as follows:

- A system variable X is considered constant, i.e., constraint $const(X)$ satisfied, if the constraint $zero(X - \overline{X})$ is satisfied within the tolerances ϵ and δ.
- To test whether Y is a derivative of X, the numerical derivative of X, $\dot{X}$ is first computed. The constraint $zero(Y - \dot{X})$ is then tested, and if satisfied within the tolerances ϵ and δ, Y is considered a derivative of X, i.e., $deriv(X, Y)$ satisfied. Note that the dimensions of X and Y (if specified) have to satisfy the constraint $dim(Y) = dim(X)/[s]$, where $[s]$ denotes time (seconds).
- The constraint $minus(X, Y)$ is tested by testing whether $zero(X+Y)$ is satisfied within the tolerances ϵ and δ. X and Y should be of the same dimension (if specified).
- To test $add(X, Y, Z)$ and $mult(X, Y, Z)$, the constraints $zero(Z - X - Y)$ and $zero(Z - X \cdot Y)$ are tested. The dimensions of the variables (if specified) have to satisfy $dim(X) = dim(Y) = dim(Z)$ for the $add(X, Y, Z)$, and $dim(Z) = dim(X) \cdot dim(Y)$ for the $mult(X, Y, Z)$ constraint.
- To test $M^+(X, Y)$, we check that for (almost) all time points t_i and t_j, $(X_{t_i} > X_{t_j} \Leftrightarrow Y_{t_i} > Y_{t_j})$. More precisely, $M^+(X, Y)$ is considered satisfied if $P(DX \cdot DY < 0/|DX| > \delta \wedge |DY| > \delta) < \epsilon$, where $DX = X_{t_i} - X_{t_j}$ and $DY = Y_{t_i} - Y_{t_j}$. Similarly, $M^-(X, Y)$ is considered satisfied if $P(DX \cdot DY > 0/|DX| > \delta \wedge |DY| > \delta) < \epsilon$. In this case, there are no dimension constraints for X and Y.

Let us now consider the computational complexity of QMN. If we let q_k denote the number of variables of depth at most k, i.e., $q_k = |V_k|$, we have $q_1 = (o + 1)n$ and $q_k \leq 4q_1q_{k-1}$, which gives $q_d = O((4n(o + 1))^d)$ variables at the end of the process of introducing new variables. The number of $const$ constraints tested is $O(q_d)$, the number of $deriv$, $minus$, M^+ and M^- constraints is $O(q_d)^2$, and the number of add and $mult$ constraints is $O(q_d)^3$.

Each of the qualitative constraints $const$, $deriv$, $minus$, add and $mult$ takes $O(N)$ to test ($N+1$ is the total number of time points in the example behavior). Namely, the calculation of the variables that are tested for representing the constant zero takes $O(N)$, and the counting necessary to test the constraint $zero$ also takes $O(N)$. The counting needed for testing M^+ and M^- takes $O(N^2)$, as there are N^2 pairs of values for the variables considered. Thus, testing M^+ and M^- takes $O(N^2)$. The total time complexity of QMN is then

$$O(N^2(4n(o + 1))^{2d} + N(4n(o + 1))^{3d}).$$

3. Experiments with QMN

The experiments with QMN proceeded as follows: A set of differential equations, modeling a real-life dynamical system was first chosen, as well as appropriate values of the parameters involved. The initial state for the system variables, the integration step h for solving the differential equations and the number N of integration steps were next selected. The differential equations were then integrated using the fourth-order Runge-Kutta method [Press, et al. 1986] (pp. 550–554). The obtained behavior was then given to QMN, which generated a set of qualitative laws. Dimensional information on the system variables was also given to QMN.

QMN was applied to two dynamical systems which have been used earlier as test examples for the automatic generation of qualitative models. These are the U-tube system and the cascaded tanks system [Kraan, et al. 1991]. The parameters ϵ and δ were set to 0.01 and 0.001. QMN is written in the C programming language and was run on a Sun SPARC IPC workstation. The running times for the two experiments described below ($d = 1$) were of the order of one minute.

3.1. *The U-tube*

The U-tube system consists of two containers, A and B, connected by a thin pipe and filled with liquid to levels l_A and l_B, respectively. The difference in the levels causes the liquid to flow through the pipe from one container to the other. The change of the liquid levels is described by the following differential equations:

$$\dot{l_A} = c(l_B - l_A) \tag{1a}$$

$$\dot{l_B} = -\dot{l_A} \tag{1b}$$

The initial state $l_A(0) = 10$, $l_B(0) = 210$ was assumed, as well as the parameter value $c = 2$. The equations were then simulated for $N = 1000$ integration steps of $h = 0.01$ time units. The resulting behavior was then given to QMN.

When only the derivatives of the new variables are introduced ($d = 1$), the following qualitative model is generated by QMN:

$$\begin{array}{lll} deriv(l_B, \dot{l_B}) & deriv(l_A, \dot{l_A}) & minus(\dot{l_A}, \dot{l_B}) \\ M^+(l_A, \dot{l_B}) & M^+(l_B, \dot{l_A}) & M^-(l_A, l_B) \\ M^-(l_A, \dot{l_A}) & M^-(l_B, \dot{l_B}) & M^-(\dot{l_A}, \dot{l_B}) \end{array}$$

When new variables are introduced ($d = 2$), 362 constraints are found to be consistent with the example behavior. These include constraints that are typically included in the qualitative model of the U-tube, such as $M^+(\dot{l_A}, l_B - l_A)$, but also other constraints that happen to be true in the particular behavior. For example, the constraint $const(l_A + l_B)$ is true for the above behavior.

In addition, various redundant constraints (that can be derived from other constraints) are generated, e.g., $const(\dot{l_A} + \dot{l_B})$ and $add(\dot{l_A} - \dot{l_B}, l_B - \dot{l_A}, \dot{l_A} + \dot{l_B})$. As these are numerous, future versions of QMN might benefit from a path-finding (graph connectedness) approach, similar to the one in MISQ [Kraan, et al. 1991]. The latter retains only a subset of constraints which connects all system variables.

3.2. *The cascaded tanks*

The cascaded tanks system has been used to illustrate the use of MISQ [Kraan, et al. 1991] for learning qualitative models of dynamical systems. It consists of two tanks (A and B), where liquid flows from the first into the second. The first tank has a constant inflow. The whole system can be described by the following model

$$i_A = c_1 \tag{2a}$$
$$\dot{l_A} = i_A - o_A \tag{2b}$$
$$\dot{l_B} = o_A - o_B \tag{2c}$$
$$o_A = c_2\sqrt{l_A} \tag{2d}$$
$$o_B = c_2\sqrt{l_B} \tag{2e}$$

where i_A is the inflow into tank A, o_A and o_B are the outflows from tanks A and B and l_A, l_B are the corresponding liquid levels. The values $c_1 = 200$ and $c_2 = 13$ were chosen for the parameters and the behavior of the system was simulated from the initial state $l_A(0) = 10000, l_B(0) = 0$ for $N = 1000$ steps of $h = 0.01$ time units.

Given the above behavior and the parameter settings $o = 0$ (no derivatives) and $d = 1$ (no new variables), QMN found the following qualitative constraints:

$$const(i_A) \quad M^+(l_A, o_A) \quad M^+(l_B, o_B)$$

The settings $o = 1, d = 1$ produces the following ten new constraints, in addition to the five *deriv* constraints and the three constraints above. Note that the constraints $add(o_A, \dot{l_A}, i_A)$ and $add(o_B, \dot{l_B}, o_A)$ are typically found in the QDE model of the cascaded tanks dynamical system.

$$\begin{array}{ccccc} const(\dot{i_A}) & M^+(\dot{l_A}, \dot{o_A}) & M^-(l_A, \dot{l_A}) & M^-(l_A, \dot{o_A}) & M^-(o_A, \dot{l_A}) \\ M^-(o_A, \dot{o_A}) & add(o_A, \dot{l_A}, i_A) & add(o_B, \dot{l_B}, o_A) & add(\dot{i_A}, \dot{o_A}, \dot{o_A}) & add(\dot{i_A}, \dot{o_B}, \dot{o_B}) \end{array}$$

For comparison, MISQ generated a subset of the above model from a numerical trace of a behavior of the cascaded tanks system [Kraan, et al. 1991]. The system variables given to MISQ are i_A, l_A, o_A, l_B, o_B, $\dot{l_A}$ and $\dot{l_B}$, as it does not introduce new variables. Note that both MISQ and QMN use dimensional information to reduce the number of constraints which are generated and tested.

The behavior trace given to MISQ was not the same as the one given to QMN. Namely, MISQ generates a qualitative model from a numerical trace by translating the numerical trace into a qualitative behavior, from which a qualitative model is then generated. However, the transformation process is rather simplistic and encounters problems with numerical precision when given the same trace as QMN.

QMN avoids problems with numerical precision and noise by using the tolerances ϵ and δ. As MISQ is implemented in Prolog, efficiency would also be a problem when dealing with the long numerical trace given to QMN.

The reason that MISQ generated a subset of the model generated by QMN is that MISQ only retains a subset of the constraints consistent with the behavior. The subset retained should be sufficient to connect all given variables. (The model gives rise to a graph in which variables are nodes and there is an arc between X and Y if X and Y appear together in a model constraint. The variables are connected if the graph corresponding to the model is connected.)

4. The LAGRANGE algorithm

While QMN generates a set of qualitative differential equations from an example behavior, LAGRANGE generates a set of ordinary (quantitative) differential equations. It may also generate algebraic equations that contain no time derivatives. An early version of LAGRANGE is described in [Džeroski &Todorovski, 1993].

The input to LAGRANGE is a behavior trace of a dynamical system, of the same form as the one given to QMN (Table 1). Dimensions for the system variables are optional. In addition, the values of five parameters have to be specified. These are: the order o of the dynamical system (the order of the highest derivative appearing in the dynamics equations), the maximum depth d of new terms introduced by combining old terms (variables), the maximum number r of independent regression variables used for generating equations, and two tolerances, t_R and t_S, used when testing equations.

The LAGRANGE algorithm consists of three main stages. Taking the set of system variables $S = \{X_1, \ldots, X_n\}$, LAGRANGE first introduces their time derivatives (up to order o). It then introduces new variables (terms) by repeatedly applying multiplication to variables from S and their time derivatives. The values of the new variables (terms) at all time points are calculated as the variables are introduced. Finally, given the set of all variables, LAGRANGE generates and tests equations by using linear regression.

The first stage introduces the time derivatives of the system variables and is identical to the corresponding stage of QMN (Section 2). During the second stage, new variables are introduced in two ways. First, for each system variable that is measured in radians, i.e., represents an angle, the corresponding sine and cosine are added to the current set of variables (line 8). Second, the variables in V are combined by multiplication to introduce as new variables all terms (products) of degree d or less (lines 9-13). Terms of depth k are gathered in V_k. The depth of a term $\prod_{i=1}^{n} {X_i}^{\alpha_i}$ is defined as $k = \sum_{i=1}^{n} \alpha_i$. For example, the term $X_1 \sin X_2$ is of depth two and the term $X_1^3 \dot{X}_2$ is of depth four.

Equations are generated and tested in the third stage (lines 15-27). Roughly speaking, each subset of V sized at most $r + 1$ is used to generate a linear equation. The term with greatest depth (complexity) is chosen as the dependent variable (ties are resolved arbitrarily) and is expressed as a linear combination of the remaining ones. The constant coefficients in the linear equation are calculated by applying linear regression (line 20). If the equation appears to be significant, i.e., the corresponding criteria exceed the prespecified thresholds, it is added to the model.

1. **LAGRANGE**(S, o, d, r, t_R, t_S):
2.   **Introduce-time-derivatives**
3. $V := S$
4.   **forall** $v \in S$ **do**
5. $v_0 := v$
6.   **for** $i := 1$ **to** o **do** $V := V \cup \{\frac{d}{dt} v_{i-1}\}$
7.   **Introduce-new-variables**
8.   **forall** $v \in S$ such that $\dim(v) = [\mathrm{rad}]$ **do** $V := V \cup \{\sin x, \cos x\}$
9. $V_1 := V$
10.   **for** $k := 2$ **to** d **do**
11. $V_k := \emptyset$
12.   **forall** $(v, u) \in V_1 \times V_{k-1}$ **do** $V_k := V_k \cup \{v \cdot u\}$
13. $V := V \cup V_k$
14.   **Generate-and-test-equations-using-linear-regression**
15. $M := \emptyset$
16.   **for** $i := 0$ **to** r **do**
17.   **forall** $R \in \mathcal{P}(V)$ such that $|R| = i + 1$ **do**
18.   Select a dependent variable $y \in R$
19. $R := R \setminus \{y\}$
20.   **Linear-Regression** $(y, R, \mathbf{c}, \sigma_c, V_R, V_S)$
21.   **if** $(V_R \geq 1 - t_R) \wedge (V_S \leq t_S)$ **then**
22. $R' := R$
23. $\mathbf{c}' := \mathbf{c}$
24.   **forall** $x \in R$ such that $|c_x| < \sigma_{c_x}$ **do** $R' := R' \setminus \{x\}$
25.   **if** $R' \neq R$ **then Linear-Regression** $(y, R', \mathbf{c}', \sigma_{c'}, V_R, V_S)$
26. $M := M \cup \{y = c'_0 + \sum_{x \in R'} c'_x x\}$
27.   **return**(M)

The **Linear-Regression** procedure, taken from [Volk, 1958], calculates the values of the coefficients $c_0, c_1, \ldots, c_m$ in the linear equation $y = c_0 + \sum_{j=1}^{m} c_j x_j$ so as to minimize the sum of squares E (calculated in line 7), i.e., to fit the measured data $(\mathbf{X}, \mathbf{y})$ as close as possible. It also calculates the deviations $\sigma_{c_0}, \sigma_{c_1}, \ldots, \sigma_{c_m}$ of these coefficients (line 8). Finally, the significance criteria are evaluated: the multiple correlation coefficient V_R and the normalized deviation V_S (lines 11-12).

Linear-Regression $(y,\{x_1,x_2,\ldots,x_m\},\ \mathbf{c},\sigma_c,V_R,V_S)$

$$\mathbf{X} := \begin{bmatrix} 1 & x_{10} & x_{20} & \ldots & x_{m0} \\ 1 & x_{11} & x_{21} & \ldots & x_{m1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_{1N} & x_{2N} & \ldots & x_{mN} \end{bmatrix}$$

$\mathbf{y} := \begin{bmatrix} y_0 & y_1 & \ldots & y_N \end{bmatrix}^T$

$\mathbf{V} := (\mathbf{X}^T\mathbf{X})^{-1}$

$\mathbf{c} = \begin{bmatrix} c_0 & c_1 & \ldots & c_m \end{bmatrix}^T := \mathbf{V}\mathbf{y}$

for $i := 0$ **to** N **do** $\hat{y}_i := c_0 + \sum_{j=1}^{m} c_j x_{ji}$

$E := \sum_{i=0}^{N} (y_i - \hat{y}_i)^2$

for $j := 0$ **to** m **do** ${\sigma_{c_j}}^2 := v_{(j+1)(j+1)} E/(N+1-m)$

$\sigma_c := \begin{bmatrix} \sigma_{c_0} & \sigma_{c_1} & \ldots & \sigma_{c_m} \end{bmatrix}^T$

$\overline{y} = (\sum_{i=0}^{N} y_i)/(N+1)$

${V_R}^2 := 1 - E/\sum_{i=0}^{N} (y_i - \overline{y})^2$

${V_S}^2 := \frac{E/(N+1)}{\overline{y}^2 + e^{-\overline{y}^2}}$

return$(\mathbf{c},\sigma_c,V_R,V_S)$

The multiple correlation coefficient V_R (usually called R [Volk, 1958]) takes values from the interval $[0,1]$. The higher V_R, the more significant the equation. An equation is significant if $V_R \geq 1 - t_R$. The default value of t_R is 0.01.

The normalized deviation V_S is defined as the ratio of the average error of approximation $\sqrt{E/(N+1)}$ and the average $\overline{y}$ of the dependent variable. The term $e^{-\overline{y}^2}$ is added to the denominator to avoid division with zero and does not change the value of V_S significantly, unless $\overline{y} \approx 0$, in which case V_S essentially equals the average error of approximation. V_S takes values from the interval $[0,\infty)$. The lower V_S, the more significant the equation. An equation is considered significant if $V_S \leq t_S$. The default value of t_S is 0.01.

If $m = 0$, i.e., the linear equation considered is of the form $y = c_0$, a perfect fit would cause a division with zero in the calculation of V_R. Thus, only V_S is used as a significance criterion in this case. Note that $V_S = 0$ implies $V_R = 1$, but not vice versa. Therefore, V_S is a more discriminative criterion.

Before a significant equation is added to the model, the independent variables x for which $|c_x| < \sigma(c_x)$ are removed from the equation (line 24 of the LAGRANGE algorithm). This may affect the coefficients of the other independent variables. Consequently, the **Linear-Regression** procedure is invoked again to calculate the new coefficients (line 25).

An analysis of the computational complexity of the LAGRANGE algorithm, in terms of the parameters n, N, o, d and r, is given below. To calculate the total number of terms used for generating equations observe that $|V_1| \leq (o+3)\cdot n$ and $|V_k| \leq |V_1||V_{k-1}|$, which gives $|V_k| = O(((o+3)n)^k)$. This yields a total number of terms $|V| = O(((o+3)n)^d)$ at the end of the second stage. The total number of regressions tried is $O(|V|^{r+1})$, i.e., $O(((o+3)n)^{d(r+1)})$. Taking into account the complexity of the linear regression procedure $O((m+1)^3 + N(m+1)^2)$, where m is the number of independent variables, and $N >> m$, the total complexity of the LAGRANGE algorithm is $O(Nr^2((o+3)n)^{d(r+1)})$. While it is exponential in the parameters d and r, we should note that small values of these parameters are usually sufficient for real dynamical systems.

5. Experiments with LAGRANGE

The experiments with LAGRANGE were performed analogously to the ones with QMN: a set of differential equations, modeling a real-life dynamical system was first chosen, as well as appropriate values of the parameters involved and the initial state for the system variables. The differential equations were then integrated using the fourth-order Runge-Kutta method [Press, et al. 1986] (pp. 550–554) for $N = 1000$ steps of $h = 0.01$ time units. The obtained behavior was then given to LAGRANGE, which generated a set of equations describing the behavior.

LAGRANGE is implemented in the C programming language and was run on a Sun SPARC IPC workstation. It was applied to several problems in the domains of fluid dynamics (the U-tube and the cascaded tanks), population growth (population dynamics) and mechanical dynamics (the inverted pendulum). Table 3 gives an overview of the experiments conducted with LAGRANGE. For each domain, the dimension of the system (number of system variables) is given, as well as the values of the parameter settings for LAGRANGE. These include the order o, the depth d, and the number of regression variables r, as well as the significance thresholds (tolerances) t_R and t_S. Finally, the number of equations generated in each domain and the corresponding running time is given.

Table 3. An overview of the experiments conducted with LAGRANGE.

Domain	Variables	Parameters					Equations	Time [s]
		o	d	r	t_R	t_S		
U-tube	2	1	1	2	0.01	0.01	6	2.67
Cascaded tanks	2	1	2	4	0.01	0.0002	3	83.75
Population dynamics	2	1	2	2	0.01	0.01	2	9.77
Inverted pendulum	2	2	3	2	0.01	0.0001	4	1484.80

The parameters o, d, and r in LAGRANGE were set to reflect the complexity of the original models, i.e., differential equations. The significance thresholds t_R and t_S were set to 0.01 as a default, except for the two cases where this produced too many equations. In these cases, the S criterion was used to distinguish among the equations, and was progressively lowered until a reasonable number of equations was obtained. Note that no additional runs of LAGRANGE are necessary when lowering the significance thresholds; it is sufficient to filter out the equations satisfying the higher, but not the lower thresholds.

5.1. The U-tube

From the behavior of the U-tube dynamical system described in Section 3.1, LAGRANGE generated the following equations:

$$
\begin{aligned}
l_B &= 210 - l_A && (3a)\\
\dot{l_A} &= 420 - 4l_A && (3b)\\
\dot{l_A} &= -420 + 4l_B && (3c)\\
\dot{l_B} &= -420 + 4l_A && (3d)\\
\dot{l_B} &= 420 - 4l_B && (3e)\\
\dot{l_B} &= -\dot{l_A} && (3f)
\end{aligned}
$$

The first equation expresses the law of mass (volume) conservation, while the last equation is the same as Equation (1b) in the original model. Equations (3b) and (3c) can be obtained from Equation (1a), by taking into account Equation (3a). Finally, Equations (3d) and (3e) can be obtained from Equations (3b) and (3c) by taking into account Equation (3f).

5.2. The cascaded tanks

While the cascaded tanks system was described by a model containing five variables (Equations (2a-2e)) in Section 3.2, two variables are essentially enough to describe this dynamical system. This is reflected in the following model, which is equivalent to the one mentioned above:

$$
\begin{aligned}
\dot{l_A} &= c_1 - c_2\sqrt{l_A} && (4a)\\
\dot{l_B} &= c_2\sqrt{l_A} - c_2\sqrt{l_B} && (4b)
\end{aligned}
$$

From the behavior described in Section 3.2, LAGRANGE generated three equations under the parameter settings given in Table 3. The t_S parameter was lowered from the default 0.01 to 0.0002 to obtain a small number of equations. Namely, the default values of t_R and t_S resulted in 474 equations. Thus, t_S was progressively lowered. The best two equations ($R = 1$ and lowest values for S) are given below.

$$
\begin{aligned}
\dot{l_A}^2 &= -40000 + 169l_A + 400\dot{l_A} && (5a)\\
\dot{l_B}^2 &= -169l_A + 169l_B + 400\dot{l_B} - 2\dot{l_A}\dot{l_B} && (5b)
\end{aligned}
$$

Equation (5a) can be obtained by expressing $\sqrt{l_A}$ from Equation (4a) and squaring both sides of the resulting equation. Equation (5b) can be obtained by adding Equations (4a) and (4b), expressing $\sqrt{l_B}$, squaring both sides of the resulting equation and finally subtracting Equation (5a).

5.3. Population dynamics

A Volterra-Lotka model of periodic behavior can be used to model the coexistence of prey and predator populations [Babloyantz, 1986] (p. 145). For example, take the populations of lynxes and hares. The latter are grazing on grass (we assume unlimited supply of grass), and the former are carnivores that hunt hares (we assume hares are their only food). If the hare population is large, the lynx population grows fast. This causes many hares to be eaten, thus diminishing the hare population to the point where there is not enough food for the lynxes. The lynx population consequently decreases, and hares can multiply much faster. This behavior is depicted in Figure 1, where N_1 denotes the size of the hare and N_2 the size of the lynx population.

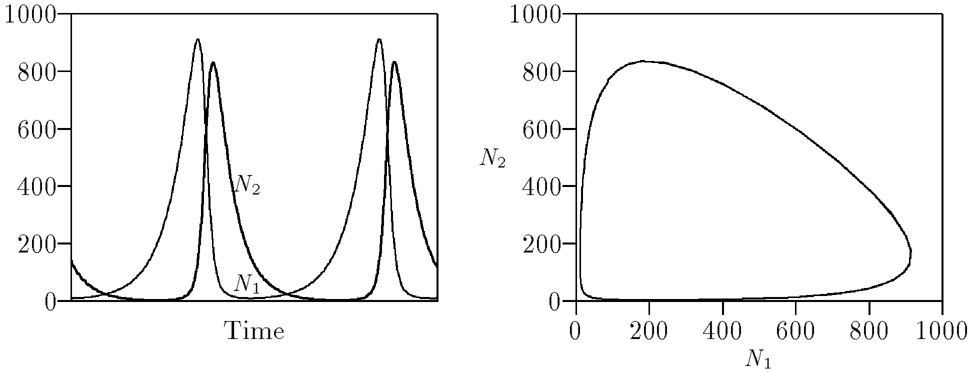

Figure 1. The oscillatory behavior of the sizes of the hare and lynx populations.

The oscillatory behavior of the sizes of the hare and lynx populations can be described by the following model:

$$\dot{N}_1 = k_1 N_1 - s N_1 N_2 \quad (6a)$$
$$\dot{N}_2 = s N_1 N_2 - k_2 N_2 \quad (6b)$$

Choosing the initial hare and lynx populations to be $N_1(0) = 10$ and $N_2(0) = 140$, the predation (s), growth (k_1) and the death (k_2) rates to be $s = 0.01$, $k_1 = 1.6$, and $k_2 = 0.2$, and integrating the differential equations for $N = 1000$ steps of $h = 0.01$ time units results in the behavior shown in Figure 1.

Given the above behavior, LAGRANGE discovered the following two equations, which can be obtained by algebraic manipulation of each of the two equations in the original model, and are thus equivalent to original model.

$$N_1 N_2 = 160 N_1 - 100 \dot{N}_1 \quad (7a)$$
$$N_1 N_2 = 200 N_2 - 100 \dot{N}_2 \quad (7b)$$

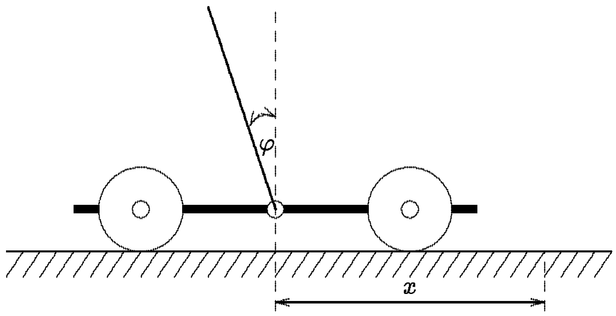

Figure 2. The inverted pendulum.

5.4. *The inverted pendulum*

The task of balancing the inverted pendulum is a standard benchmark problem for testing classical and modern approaches to the control of dynamical systems [Geva & Sitte, 1993; Urbančič & Bratko, 1994]. The inverted pendulum consists of a cart that can move along a horizontal track, and a pole hinged on top of the cart, so that it can rotate in the vertical plane defined by the track and its fixed point (Figure 2). A force parallel to the track can be applied to the cart.

The system variables in this case are x, the distance of the cart from the origin point on the track, and φ, the inclination angle of the pole relative to the vertical line through its fixed point (Figure 2). The dynamics of the inverted pendulum can be described by a system of second order differential equations. Although the explicit form of the equations (expressing $\ddot{x}$ and $\ddot{\varphi}$) looks quite complicated, the equivalent implicit form listed below is quite manageable. In the equations, M and m are the masses of the cart and the pole, respectively, l is the length of the pole, F the force applied to the cart, and $g = 9.81$ is the gravitational acceleration.

$$(M+m)\ddot{x} + \frac{1}{2}ml(\ddot{\varphi}\cos\varphi - \dot{\varphi}^2\sin\varphi) = F \tag{8a}$$

$$\ddot{x}\cos\varphi + \frac{2}{3}l\ddot{\varphi} = g\sin\varphi. \tag{8b}$$

Taking $M = 1[kg]$, $m = 0.1[kg]$, and $l = 1[m]$, the following behavior was simulated. Initially, the cart is at the origin and the pole is inclined to the left ($x(0) = 0, \varphi(0) = 3\pi/16$), both being immobile ($\dot{x}(0) = 0, \dot{\varphi}(0) = 0$). The cart is then pushed to the left with a constant force of $F = 7.5[N]$. This results in the pole passing through the vertical position and finally falling down to the right, while the cart is moving left all the time. The pole falls down after $N = 133$ integration steps of $h = 0.01$ seconds.

In this experiment, the quality criterion S was used to discriminate among the equations considered. The threshold value t_S was set to $t_S = 10^{-4}$, as the default parameter values for t_R and t_S resulted in a large number of equations. The value of t_S was progressively lowered in several trials, until the number of equations produced by LAGRANGE was manageable.

$$\cos^2\varphi = 1-\sin^2\varphi \tag{9a}$$
$$\ddot{x}\cos\varphi = -0.67\ddot{\varphi}+9.81\sin\varphi \tag{9b}$$
$$\dot{\varphi}^2\sin\varphi = -150+22\ddot{x}+\ddot{\varphi}\cos\varphi \tag{9c}$$
$$\ddot{x}\cos\varphi\sin\varphi = 9.81-0.67\ddot{\varphi}\sin\varphi-9.81\cos^2\varphi \tag{9d}$$

Given the time series for x, φ, $\dot{x}$, and $\dot{\varphi}$, and the settings described above, LAGRANGE discovered Equations (9a-9d). The first equation expresses the known relationship between the sine and the cosine of an angle. Equations (9b) and (9c) are equivalent to the original Equations (8b) and (8a), respectively. Finally, Equation (9d) can be obtained by multiplying Equation (8b) by $\sin\varphi$ and taking into account Equation (9a).

6. Related work

The QMN and LAGRANGE systems, described above, are related to several areas in the fields of qualitative reasoning and machine learning. These include automatic generation of qualitative models, inductive logic programming and machine discovery. They are also related to work on identification of dynamical systems.

Both LAGRANGE and QMN use a generate and test approach, similar to existing systems for automatic generation of qualitative models from example behaviors, such as GENMODEL [Coiera, 1989] and MISQ [Kraan, et al. 1991]. In addition, QMN addresses precisely the same problem as MISQ, namely the problem of generating qualitative models from numerical behavior traces. Finally, QMN was preceded by attempts to learn QDE [Kuipers, 1986] models of dynamical systems [Bratko, et al. 1992; Džeroski & Bratko, 1992], which relied on inductive logic programming [Muggleton, 1992; Lavrač & Džeroski, 1994] systems to induce models from example qualitative behaviors and background knowledge consisting of logical definitions of the QDE constraints.

QMN and LAGRANGE are inspired and benefit from ideas developed in the field of inductive logic programming in several ways. First of all, QMN addresses a problem that has also been tackled by the GOLEM [Bratko, et al. 1992] and mFOIL [Džeroski & Bratko, 1992] systems for inductive logic programming. Second, both QMN and LAGRANGE introduce new variables in the same way as new variables are introduced by determinate literals in LINUS [Džeroski, et al. 1992]. Finally, they are also related to CLAUDIEN [De Raedt & Bruynooghe, 1993], which performs clausal discovery by systematically generating clausal integrity constraints and testing them for consistency with a given database.

Following the LINUS approach to inductive logic programming [Lavrač, et al. 1991; Džeroski, et al. 1992], LAGRANGE was initially to introduce as new variables the time derivatives of the given system variables and then apply existing machine discovery systems on the transformed problem. However, two basic requirements have to be satisfied by a machine discovery system to be used on the transformed discovery problem. First, it must be able to find laws involving more than two variables from observational data only, i.e. without asking for additional experiments. Second, it must be able to find a set of laws rather than a single one, where all laws hold for the domain as a whole and it is not explicitly stated beforehand which of the variables involved are dependent and which independent.

Existing discovery systems, such as BACON [Langley, et al. 1987], ABACUS [Falkenheiner & Michalski, 1990] and FAHRENHEIT [Żytkow & Zhu, 1991], satisfy the first or second criterion, but none of them meets both criteria in a satisfactory way. Systems like BACON and FAHRENHEIT typically propose experiments and require the outcomes to be provided, thus violating the first criterion. Regarding the second criterion, ABACUS and FAHRENHEIT are able to discover more than one law in a domain. However, these laws usually hold in disjoint subspaces of the domain. There have also been some ideas about discovering a set of simple laws from observational data with BACON, but they have not actually been incorporated into BACON. Thus, none of the above systems meets both of the above criteria in a satisfactory way.

The Equation Finder [Zembowicz & Żytkow, 1992], used as a module in FAHRENHEIT, comes closest to meeting the above requirements. Given a set of real-valued pairs (x_i, y_i), it tries to find a formula $Y = f(X)$, without asking for additional data. The main problem with Equation Finder is that it is looking for a function of one variable only. In dynamical systems, however, the value of a system variable at a certain point in time can depend on the present and past values of several system variables. This makes the ability to discover equations with more than two variables a necessity.

One of the important features of Equation Finder is the ability to handle errors in the input data. Such a feature is necessary for handling real-life data. It is also important that the convergence of Equation Finder and its sensitivity to errors in the input data have been thoroughly studied [Zembowicz & Zytkow, 1992].

Although the machine discovery community has not paid much attention to dynamical systems, it has recently recognized the need for techniques that can handle such systems [Bielecki, 1992]. Also, the IDS [Nordhausen & Langley, 1990] system touches upon the problem of discovering dynamics. The qualitative schemata generated by IDS can be considered qualitative states of a system. The transitions between states (schemata) capture in a way the qualitative change of the system over time, i.e., its global dynamics. They are more of the phase transition type, rather than continuous state changes as described by differential equations. However, numerical laws within the qualitative states could, in principle, include time-dependent variables.

Finally, QMN and LAGRANGE are related to work in the area of dynamical system identification, which addresses essentially the same problem as LAGRANGE. In mainstream system identification, as summarized by Ljung [Ljung, 1993], the assumption is that the model structure, i.e., the form of the differential equations is known. The task is then to determine suitable values for the parameters appearing in the model, so that the model fits the observed behavior. Linear model structures are most often used. In contrast, QMN and LAGRANGE do not assume a prespecified model structure, but rather explore a space of equations (which can be nonlinear) and report those that are consistent with the observed behavior.

7. Discussion

The most important contribution of our work is the extension of the scope of machine discovery to dynamical systems. Namely, QMN and LAGRANGE are able to construct a set of qualitative, respectively ordinary, differential equations describing a given behavior of a dynamical system. In this way, they extend the scope of machine discovery systems from high-school physics to college physics.

It is also important that QMN and LAGRANGE are able to generate a set of laws, involving more than two variables, from observational data only. This is in contrast with some machine discovery systems, which perform problem decomposition to find laws involving more than two variables. In practice, these systems keep some variables constant and require experiments that vary the others. This is impossible in many cases, especially in the context of dynamical systems, as the scientist might not have full experimental control over all the system variables.

QMN brings several improvements over existing approaches to automatic generation of qualitative models. First, it can introduce new variables, while GENMODEL [Coiera, 1989] and MISQ [Kraan, et al. 1991] cannot. Second, QMN can handle numerical precision and noise problems through the built-in tolerances, which MISQ cannot handle. Finally, QMN is more efficient than MISQ. The inductive logic programming approaches to automatic generation of qualitative models also have problems with the introduction of new variables [Bratko, et al. 1992; Džeroski & Bratko, 1992]. Namely, new variables introduced by adding existing qualitative variables do not have uniquely determined values, i.e., are indeterminate, and cause computational complexity problems. One of our original motivations for the development of QMN was to avoid the problem of indeterminacy (adding quantitatively valued system variables yields unique outcomes). We consider QMN to be a stepping stone from the area of learning qualitative models, which intersects inductive logic programming, to the area of machine discovery, where LAGRANGE is situated.

LAGRANGE is unique among machine discovery systems in its ability to discover laws that govern the behavior of dynamical systems. Although it might be regarded as a small extension of existing discovery algorithms, it can handle a very large class of new problems. Our work on LAGRANGE is very much in the BACON tradition [Langley, et al. 1987], showing that one can handle a large number of apparently complex laws with very simple discovery mechanisms. This is demonstrated by the successful use of LAGRANGE for generating models of several dynamical systems from simulated data, including two fluid dynamics models, a population dynamics model, and the inverted pendulum model. In all cases, LAGRANGE generated models correctly describing the given behaviors. Judging on the successful use of LAGRANGE in building models of dynamical systems, we conclude that LAGRANGE is a promising step towards the application of machine discovery to real dynamical systems. As the task of identification of dynamical systems (discovering dynamics) is omnipresent, LAGRANGE is potentially applicable to a wide variety of real-life problems.

Several directions for further work appear promising at present. The problem of handling noise and measurement errors has to be addressed first. Extending the space of models/equations explored promises wider applicability, but further increases the computational complexity. In this respect, the use of domain specific knowledge has to be considered. Finally, the integration of both logical conditions and numerical laws in the discovered models deserves attention from both the inductive logic programming and the dynamical system identification perspective.

The highest priority in improving LAGRANGE is the handling of measurement errors and noise in the input data, which is necessary for practical applications. Experiments on the domains presented in this paper with artificially introduced noise have shown that LAGRANGE is very sensitive to noisy data [Križman, 1994]. The numerical derivation process has been identified as the main cause of this problem. To alleviate it, the GOLDHORN system [Križman, 1994], a descendant of LAGRANGE, employs two remedies: digital filtering and numerical integration. An appropriately different fitting mechanism, the downhill simplex method [Press, et al. 1986] is used to determine the parameter values in the equations. GOLDHORN also searches for differential equations that explicitly express the highest derivatives in terms of the system variables and their derivatives of lower order. In this way, it sidesteps the redundancy problem present in LAGRANGE. Finally, let us mention that, in addition to successfully reconstructing dynamical system models from noisy data, GOLDHORN has been successfully applied to two real-life domains.

Although at present new terms are introduced by multiplication only, other transformations, such as those used in Equation Finder could be easily incorporated within LAGRANGE. The number of new terms would significantly increase in this case and would require reduction of the search complexity. Heuristics should thus be used to reduce the number of new terms introduced. In addition, the terms for linear regressions should be chosen in a more intelligent way. Depending on the transformations used, the linear regression procedure may become unsuitable for determining the appropriate values of the equation parameters. This is the case when the equations considered are nonlinear in their parameters. Nonlinear optimization, which is computationally expensive, has to be used for such cases.

An alternative approach could be the use of genetic search techniques, which would not systematically generate and test all possible equations, but stochastically explore a subset of the space of possible equations. This would allow for reasonable time complexity even in cases where the space of possible equations is much larger than the one exploited by LAGRANGE. Džeroski and Petrovski [1994] report on preliminary experiments along the above lines. Their approach can also use domain specific knowledge in the form of substructures (subexpressions) that are likely to appear in the model of the dynamical system. This is a compromise between completely specifying the model structure, as in mainstream system identification, and not specifying any structure, as in LAGRANGE.

The inductive logic programming system CLAUDIEN [De Raedt & Bruynooghe, 1993] systematically generates logical constraints and tests them for consistency with a given database. This is very similar to the approach taken in LAGRANGE, which systematically generates equations and tests them for consistency with a given behavior. The similarity suggests the possibility of integration [Van Laer & De Raedt, 1993]. Adding the arithmetic predicate $mult(X, Y, Z)$ as background knowledge in CLAUDIEN, as well as a predicate implementing linear regression, CLAUDIEN would encompass both the numerical law capabilities of LAGRANGE and the ability to use logical conditions in the generated models. Recent research in the area of dynamical system identification [Ljung, 1993] suggests that such models are of great practical interest.

Acknowledgements

The research described in this paper has been funded by the Slovenian Ministry of Science and Technology and in part by the ESPRIT III Basic Research Project No.6020 Inductive Logic Programming. This paper was written during the visit of Sašo Džeroski to the Katholieke Universiteit Leuven, Belgium, supported by the Commission of the European Communities under grant CIPA3510920370. We would like to thank Ivan Bratko, Luc De Raedt, Djani Juričić, Boris Kompare, Pat Langley, Bojan Orel, Tanja Urbančič and Jan Żytkow for the stimulating discussions and suggestions regarding the work described in this paper.

References

[1] Babloyantz, A. (1986). *Molecules, Dynamics, and Life.* John Wiley & Sons, New York.

[2] Bielecki, M. W. (1992). Machine discovery approach to dynamic systems in the real laboratory. *ML92 Workshop on Machine Discovery.* Aberdeen, Scotland.

[3] Bohte, Z. (1991). *Numerical Methods.* The Society of Mathematicians, Physicists and Astronomers of Slovenia, Ljubljana, Slovenia.

[4] Bratko, I., Muggleton, S., and Varšek, A. (1992). Learning qualitative models of dynamic systems. In Muggleton, S., editor, *Inductive Logic Programming*, pages 437–452. Academic Press, London.

[5] Coiera, E. (1989). Learning qualitative models from example behaviors. *Third International Workshop on Qualitative Physics.* Stanford, CA.

[6] De Raedt, L. and Bruynooghe, M. (1993). A theory of clausal discovery. In *Proc. Thirteenth International Joint Conference on Artificial Intelligence*, pages 1058–1063. Morgan Kaufmann, San Mateo, CA.

[7] Džeroski, S. and Bratko, I. (1992). Handling noise in inductive logic programming. *Second International Workshop on Inductive Logic Programming.* Tokyo, Japan.

[8] Džeroski, S. and Petrovski, I. (1994). Discovering dynamics with genetic programming. In *Proc. Seventh European Conference on Machine Learning.* Springer, Berlin. To appear.

[9] Džeroski, S. and Todorovski, L. (1993). Discovering dynamics. In *Proc. Tenth International Conference on Machine Learning*, pages 97–103. Morgan Kaufmann, San Mateo, CA, 1993.

[10] Džeroski, S., Muggleton, S., and Russell, S. (1992). PAC-learnability of determinate logic programs. In *Proc. Fifth ACM Workshop on Computational Learning Theory*, pages 128–135. ACM Press, New York.

[11] Falkenheiner, B. and Michalski, R. (1990). Integrating quantitative and qualitative discovery in the ABACUS system. In Kodratoff, Y. and Michalski, R., editors, *Machine Learning: An Artificial Intelligence Approach*, pages 153–190. Morgan Kaufmann, San Mateo, CA.

[12] Geva, S. and Sitte, J. (1993). A cartpole experimental benchmark for trainable controllers. *IEEE Control Systems*, 13(5):40–51.

[13] Kraan, I., Richards, B., and Kuipers, B. (1991). Automatic abduction of qualitative models. *Fifth International Workshop on Qualitative Physics*. Austin, TX.

[14] Križman, V. (1994) Handling noisy data in automated modeling of dynamical systems. MSc Thesis, Faculty of Electrical and Computer Engineering, University of Ljubljana, Slovenia.

[15] Kuipers, B. (1986). Qualitative simulation. *Artificial Intelligence*, 29(3):289–338.

[16] Langley, P., Simon, H., Bradshaw, G., and Żytkow, J. (1987). *Scientific discovery.* MIT Press, Cambridge, MA.

[17] Lavrač, N. and Džeroski, S. (1994) *Inductive Logic Programming: Techniques and Applications.* Ellis Horwood, Chichester.

[18] Lavrač, N., Džeroski, S., and Grobelnik, M. (1991). Learning nonrecursive definitions of relations with LINUS. In *Proc. Fifth European Working Session on Learning*, pages 265–281. Springer, Berlin.

[19] Ljung, L. (1993). Modelling of industrial systems. In *Proc. Seventh International Symposium on Methodologies for Intelligent Systems*, pages 338–349. Springer, Berlin.

[20] Muggleton, S., editor (1992). *Inductive Logic Programming.* Academic Press, London.

[21] Nordhausen, B. and Langley, P. (1990). A robust approach to numeric discovery. In *Proc. Seventh International Conference on Machine Learning*, pages 411–418. Morgan Kaufmann, San Mateo, CA.

[22] Press, W. H., Flannery, B. P., Teukolsky, S. A., and Vetterling, W. T. (1986). *Numerical Recipes.* Cambridge University Press, Cambridge, MA.

[23] Urbančič, T. and Bratko, I. (1994) Learning to control dynamic systems. In Michie, D., Spiegelhalter, D., and Taylor, C., editors, *Machine Learning, Neural and Statistical Classification.* Ellis Horwood, Chichester. In press.

[24] Van Laer, W. and De Raedt, L. (1993). Discovering quantitative laws in inductive logic programming. *MLnet Workshop on Machine Discovery*. Blanes, Spain.

[25] Volk, W. (1958). *Applied Statistics for Engineers.* McGraw-Hill, New York.

[26] Zembowicz, R. and Żytkow, J. (1992). Discovery of equations: experimental evaluation of convergence. In *Proc. Tenth National Conference on Artificial Intelligence*, pages 70–75. MIT Press, Cambridge, MA.

[27] Żytkow, J. and Zhu, J. (1991). Application of empirical discovery in knowledge acquisition. In *Proc. Fifth European Working Session on Learning*, pages 101–117. Springer, Berlin.